Basic statistics in R

Alessio Palmisano

1. Descriptive statistics

Before starting, we will explore how to import data from tables, as these are among the most commonly used data sources in daily routine.

Spreadsheet data from programs like Microsoft Excel or OpenOffice Calc can be converted into CSV (Comma-Separated Values) files and then imported into R as data frame objects.

1. Open the spreadsheet "spearheads.xls" stored in the folder "class 2" using a program such as Microsoft Excel or OpenOffice Calc. This dataset, slightly modified from the one used in the introductory manual by Fletcher and Lock (2005), contains data on 40 bronze and iron spearheads.

2. The first column (**num**) represents a unique identifier for each spearhead. The next two variables, **mat** (Material) and **con** (Context), are categorical. **loo** (Loop) and **peg** (Peghole) are logical variables (TRUE or FALSE). The **cond** (Condition) variable is ordinal, with values ranging from 1 (Excellent) to 4 (Poor). The remaining variables are numerical and are described in Figure 1.

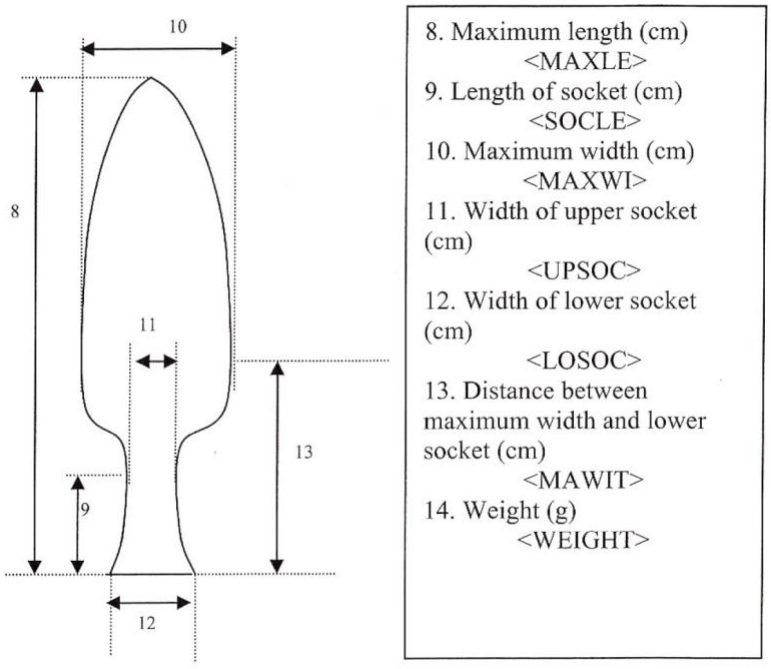


Figure 1. Numerical variables in the spearhead dataset (Fletcher and Lock 2005, p.5)

3. To save the file as a CSV, navigate to the File menu, select Save As, choose CSV as the file format, and save it in the "class 2" folder.

4. Now, be sure that you are working in the right working directory. To set it click on Session->Set working directory->Choose directory…

5. To check the current working directory, use:

```
getwd()
```

6. Now, we will import the CSV file into R as a data.frame object using the following command:

```
spearheads <- read.csv(file = "spearheads.csv", header = TRUE, sep = ",")
```

7. This creates a new data.frame object named *spearheads*.

8. To view the contents of *spearheads*, simply type in the console its name and press Enter. You will notice that any blank values from the original Excel file have been automatically stored as NA in R. Alternatively, you can left-click on the icon spearheads under the tab **Environment** in the right upper pane of RStudio.

Now, we will use R to perform a basic exploratory data analysis on the *spearheads* data frame, which we previously imported from a spreadsheet program.

The key summary statistics can be obtained using the summary() function:

```
summary(spearheads)
```

```
##       num            mat                 con                 loo        
##  Min.   : 1.00   Length:40          Length:40          Min.   :0.000  
##  1st Qu.:10.75   Class :character   Class :character   1st Qu.:0.000  
##  Median :20.50   Mode  :character   Mode  :character   Median :0.000  
##  Mean   :20.50                                         Mean   :0.275  
##  3rd Qu.:30.25                                         3rd Qu.:1.000  
##  Max.   :40.00                                         Max.   :1.000  
##                                                                       
##       peg              cond            date          maxle       
##  Min.   :0.0000   Min.   :1.000   Min.   :  50   Min.   :10.20  
##  1st Qu.:0.0000   1st Qu.:2.000   1st Qu.: 350   1st Qu.:13.50  
##  Median :1.0000   Median :2.000   Median : 650   Median :17.80  
##  Mean   :0.6923   Mean   :2.275   Mean   : 635   Mean   :20.67  
##  3rd Qu.:1.0000   3rd Qu.:3.000   3rd Qu.: 825   3rd Qu.:24.18  
##  Max.   :1.0000   Max.   :4.000   Max.   :1200   Max.   :72.40  
##  NA's   :1                                       NA's   :2      
##      socle           maxwi           upsoc           losoc       
##  Min.   : 2.400   Min.   :1.800   Min.   :0.800   Min.   :1.500  
##  1st Qu.: 4.250   1st Qu.:3.050   1st Qu.:1.300   1st Qu.:1.700  
##  Median : 5.500   Median :4.100   Median :1.500   Median :2.000  
##  Mean   : 6.141   Mean   :4.187   Mean   :1.535   Mean   :2.053  
##  3rd Qu.: 7.900   3rd Qu.:5.500   3rd Qu.:1.700   3rd Qu.:2.400  
##  Max.   :14.400   Max.   :6.400   Max.   :2.200   Max.   :2.700
```

```
##  NA's   :1        NA's   :1
##      mawit            weight
##  Min.   : 5.200   Min.   :  67.7
##  1st Qu.: 6.150   1st Qu.: 178.2
##  Median : 8.400   Median : 309.0
##  Mean   : 9.236   Mean   : 442.4
##  3rd Qu.:11.200   3rd Qu.: 559.4
##  Max.   :18.100   Max.   :2446.5
##  NA's   :1
```

This command provides essential statistical measures for the variables in the spearheads dataset, including the minimum, maximum, median, mean, first quartile, and third quartile.

Notice that for nominal variables, R outputs only the count of each category, whereas ordinal variables are treated as numerical.

Let us now perform the summary statistics for one of our numerical variables. For example, the maximum length:

```
summary(spearheads$maxle)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   10.20   13.50   17.80   20.67   24.18   72.40       2
```

To examine the distribution's spread, we can calculate its standard deviation using the following command:

```
sd(spearheads$maxle)
```

```
## [1] NA
```

Notice that if we compute the sd on the column maxle, the results will be NA. This is because the variable has some NA members which are included in the calculation. To avoid this, and remove the NAs many R functions have the na.rm parameter which, when set to TRUE, can remove the NAs from calculation. For instance:

```
sd(spearheads$maxle, na.rm = TRUE)
```

```
## [1] 11.43621
```

The standard deviation is a summary statistic that tells you, on average, how far data points are from the mean.

Now let us calculate the mean of our dataset:

```
mean(spearheads$maxle, na.rm = TRUE)
```

```
## [1] 20.67368
```

Now let us calculate the Mean ± 1 Standard Deviation

```
# Calculate mean and standard deviation
mean_value <- mean(spearheads$maxle, na.rm = TRUE)
```

```r
sd_value <- sd(spearheads$maxle, na.rm = TRUE)

# One standard deviation range
lower_bound <- mean_value - sd_value
upper_bound <- mean_value + sd_value
```

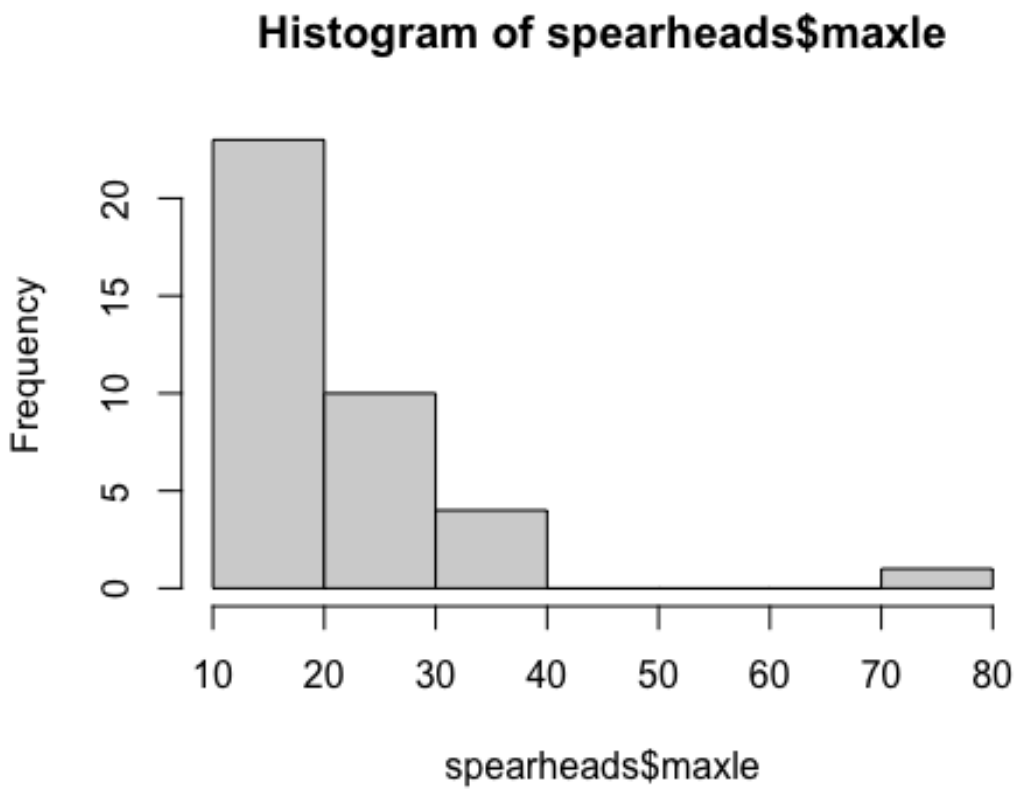
It means that 68.2% of the spearheads length is between 9.23 and 32.10 cm.

2. Plotting in R

2.1 Histograms

In R, you can easily create a histogram using the following function:

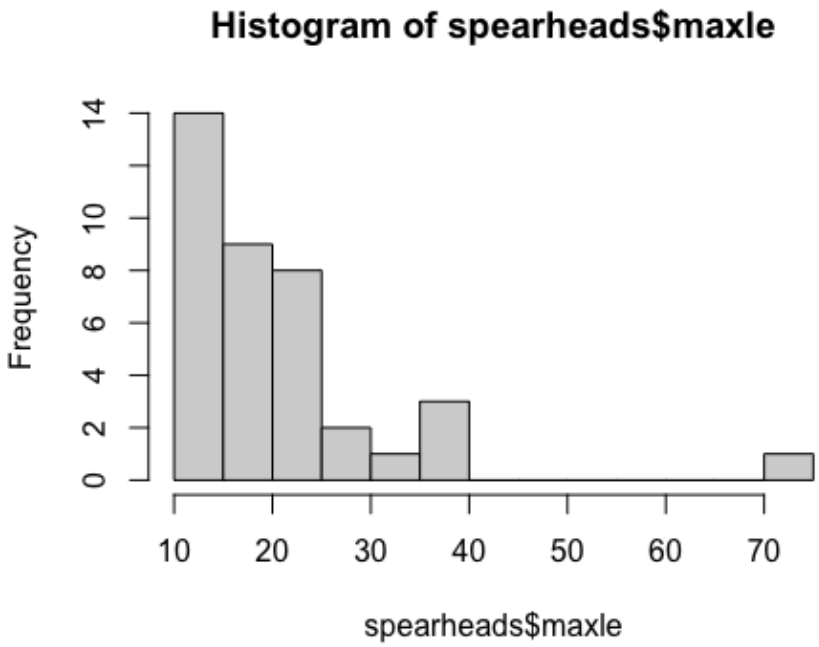
```r
hist(spearheads$maxle)
```



Notice that the hist() function automatically selects the bin size. If you'd like to modify this, you can manually specify the break values by providing a single number that indicates the desired number of bins.

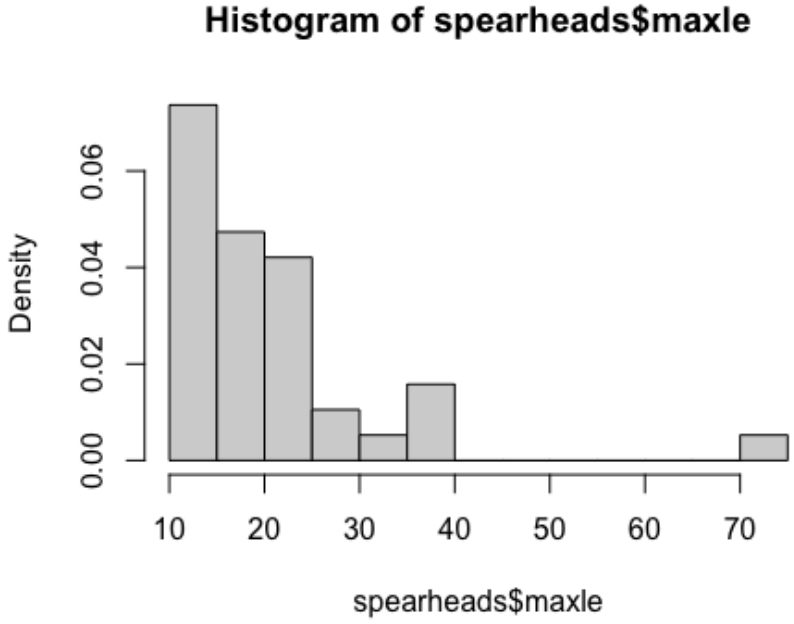
Let us try:

```r
hist(spearheads$maxle, breaks=10 )
```



Histograms can also be displayed as probabilities rather than frequencies. In this case, the height of each bin represents the probability that an observation falls within a specific interval. To do this, you can set the probability argument to TRUE:

```
hist(spearheads$maxle, breaks = 10, prob = TRUE)
```

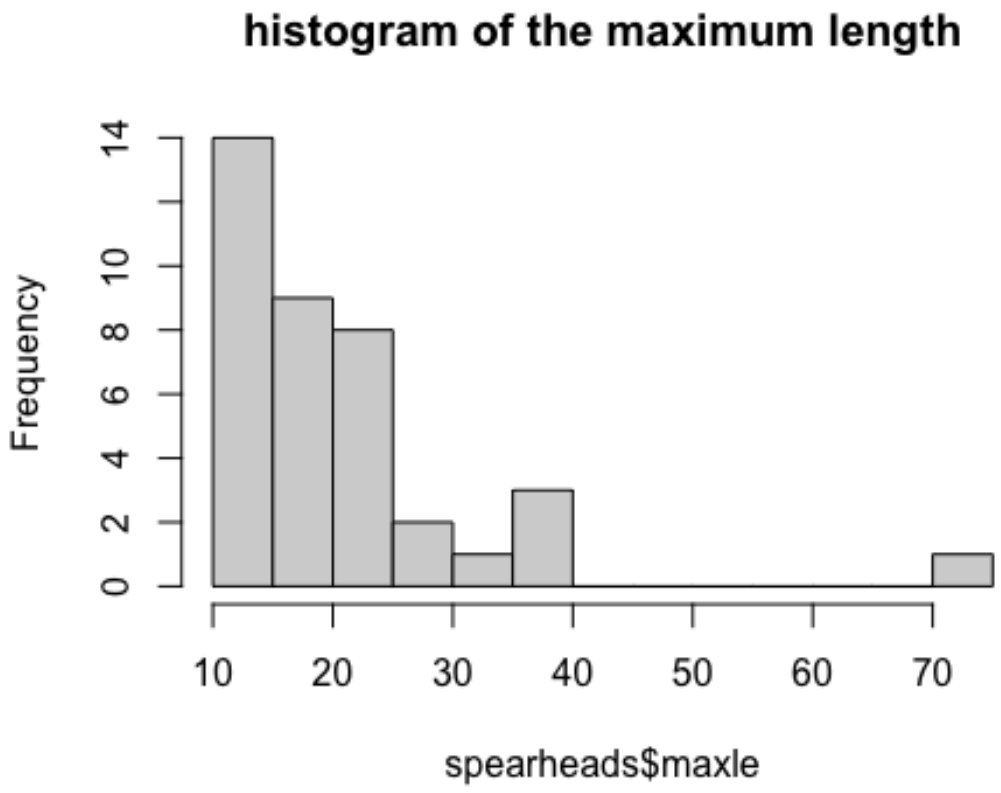


We can also add a density estimate line which will provide additional information about the shape of the distribution:

```
lines(density(spearheads$maxle,na.rm=TRUE))
```

You can also edit the title of the histogram by using the parameter main() as follows:

```
hist(spearheads$maxle, breaks = 10, main = "histogram of the maximum length")
```



Plots can be saved using the dev.print() function as follows:

```
dev.print(device=pdf, file="plot1.pdf")
```

Alternatively you can save the file as a .png:

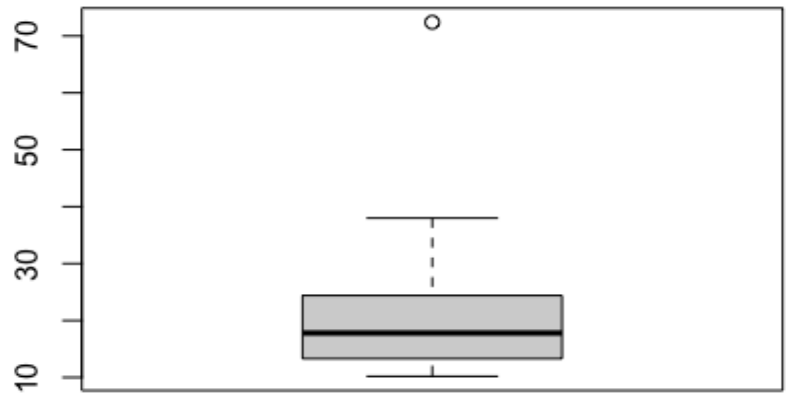
```
dev.print(device=png, file="plot1.png", width=1200, height=900, res=300)
```

2.2 Box and Whiskers Plots

A different approach to visualizing histograms is by incorporating summary statistics directly into the plot. One effective method for this is the Box and Whisker Plot, commonly known as a Boxplot, which provides a clear representation of statistical distribution.

Try the following command:

```r
boxplot(spearheads$maxle)
```



Boxplots are constructed using the median along with the upper and lower hinges, which correspond to the first and third quartiles. Horizontal lines, known as whiskers, extend to the largest and smallest observations that are not considered outliers (i.e., within 1.5 times the interquartile range). Any data points beyond this range are plotted separately, representing the extreme values in the dataset.

let us check the summary for the maximum length of our spearheads:

```r
summary(spearheads$maxle)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   10.20   13.50   17.80   20.67   24.18   72.40       2
```

The boxplot includes all observations between 13.50 cm and 24.18 cm. We have just an outlier measuring 72.40 cm.

The mean possesses unique characteristics that make it a valuable measure of central tendency. However, the presence of outliers can significantly distort its accuracy, leading to misleading conclusions. To improve reliability, removing outliers would be beneficial.The *trimmed mean* systematically removes extreme values from both upper and lower ends of a batch in a balanced fashion. In considering the level of a batch, it is the central bunch of numbers that matters most.The trimmed mean effectively avoids such confusion by simply eliminating some proportion of the highest and lowest numbers in the batch from consideration.

For instance, we could compute a 5% trimmed mean for the maximum length of our spearheads. This method involves removing the top 5% and bottom 5% of values from the dataset before calculating the mean, helping to reduce the influence of extreme outliers.

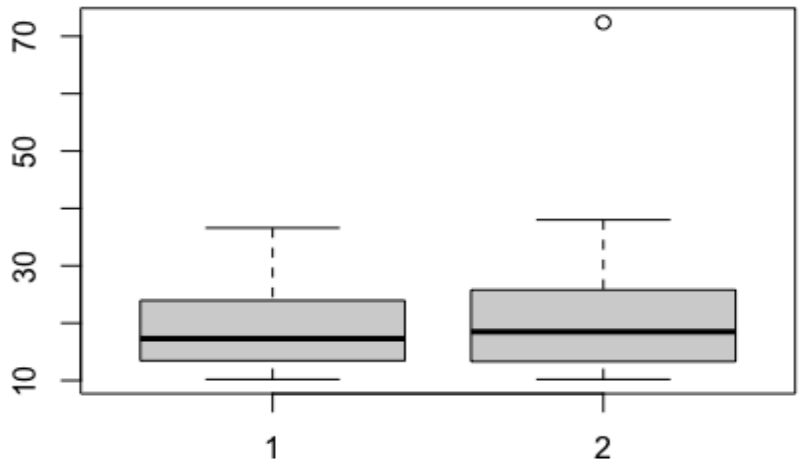
Try:

```
mean(spearheads$maxle, na.rm = TRUE,trim = 0.5)

## [1] 17.8
```

You can see that now the *trimmed mean* matches the median of our dataset computed above by using the function summary().

Boxplots are particularly effective for comparing multiple distributions. In this case, we will examine the maximum length of the spearheads (maxle) in Iron and Bronze. To achieve this, we first need to create two subsets of the variable maxle based on the material (mat).

```
boxplot(spearheads$maxle[spearheads$mat==c("Bronze")],spearheads$maxle[spearh
eads$mat==c("Iron")])
```



The plot above show that there is no much difference between the spearheads in bronze and Iron.

We can also compare the statistics bewteen the two sets:

```
summary(spearheads$maxle[spearheads$mat==c("Bronze")])

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   10.20   13.78   17.30   18.68   23.73   36.60

summary(spearheads$maxle[spearheads$mat==c("Iron")])

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   10.20   13.50   18.50   22.89   25.00   72.40       2
```

Another way to plot the distribution of values is via the violin plots. Violin plots are a variant on box- and- whiskers plots that replace the box with a kernel density plot. The violin plot produces a symmetrical kernel density distribution truncated at the minimum and maximum values. Within the distribution, a thick line represents the first and third quartiles, while the median is marked with a dot.

For plotting violin plots, we need to install the package *beanplot*

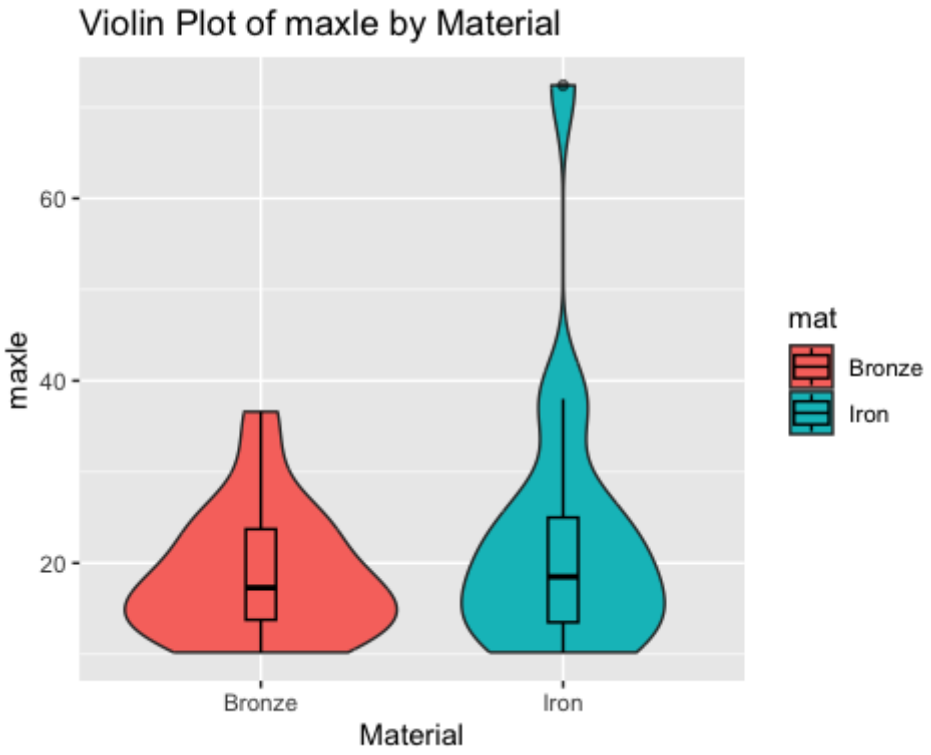
```
install.packages("ggplot2")
```

load the package

```
library(ggplot2)
```

plot the violin plots for Iron and Bronze spearheads:

```
ggplot(spearheads, aes(x = mat, y = maxle, fill = mat)) +
geom_violin() + # Violin plot
geom_boxplot(width = 0.1, color = "black", alpha = 0.5) + # Adds boxplot
inside
labs(title = "Violin Plot of maxle by Material", x = "Material", y = "maxle")

## Warning: Removed 2 rows containing non-finite outside the scale range
## (`stat_ydensity()`).

## Warning: Removed 2 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```



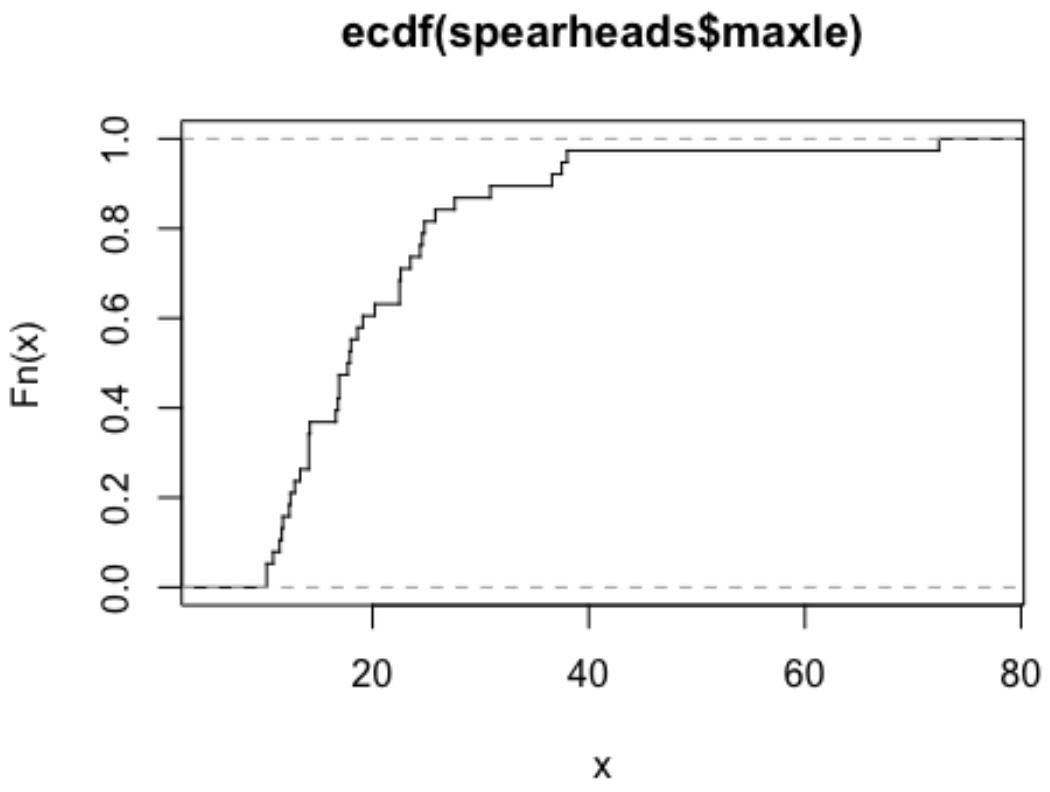
The “violin” shape itself represents the distribution of the data, with a smoothed density estimate (often using kernel density estimation, or KDE). The wider sections of the “violin” indicate higher density (more data points in that region), while the narrower sections indicate lower density.

2.3 Empirical Cumulative Frequency Distribution

The final method we introduce for visually analyzing a distribution is the cumulative frequency distribution. This approach represents the probability that a specific value in a dataset is equal to or less than other values within the dataset. If you are familiar with histograms of frequency distributions, this is essentially a cumulative version, commonly used in statistical analysis.

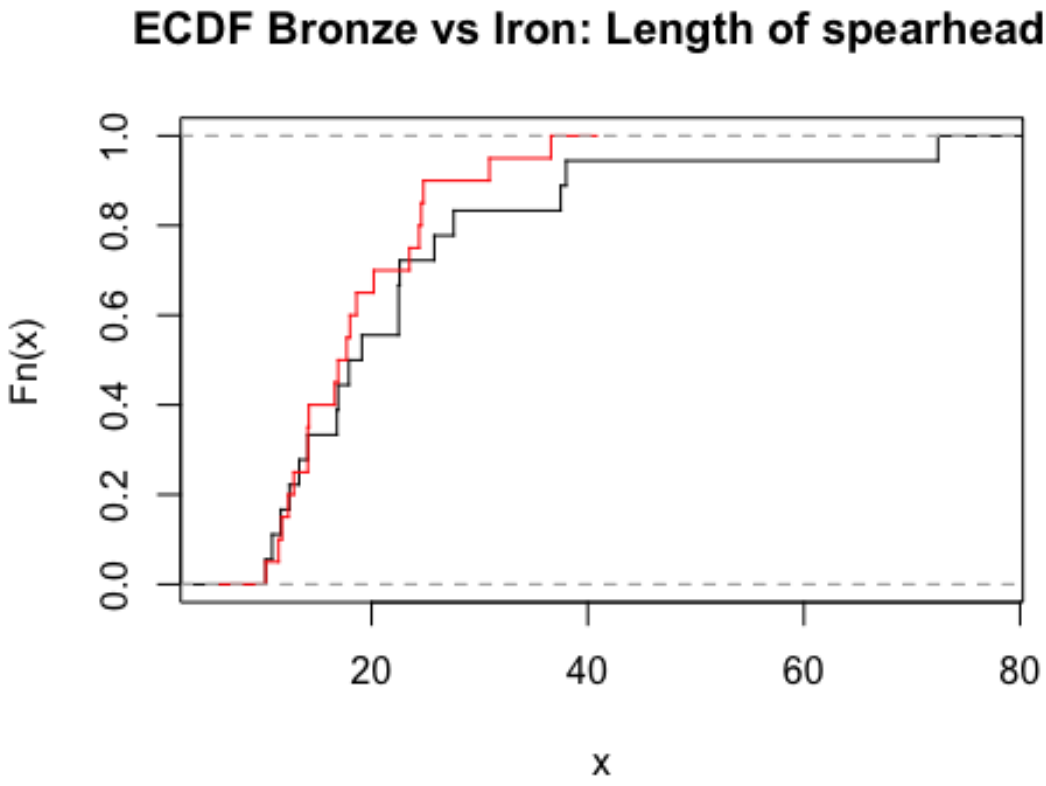
In R, this can be easily computed using the `ecdf()` function, which also allows for straightforward visualization. The following command demonstrates this process by performing two steps in one: first, generating the empirical cumulative frequency distribution from the data, and second, plotting the results.

```
plot(ecdf(spearheads$maxle), verticals=TRUE, do.points=FALSE)
```



We can also add another distribution so that we can actually compare the shape of the two distributions:

```
plot(ecdf(spearheads$maxle[spearheads$mat==c("Iron")]), verticals=TRUE,
do.points=FALSE,main="ECDF Bronze vs Iron: Length of spearheads")
lines(ecdf(spearheads$maxle[spearheads$mat==c("Bronze")]),verticals=TRUE,
do.points=FALSE,col.hor="red",col.ver="red")
```

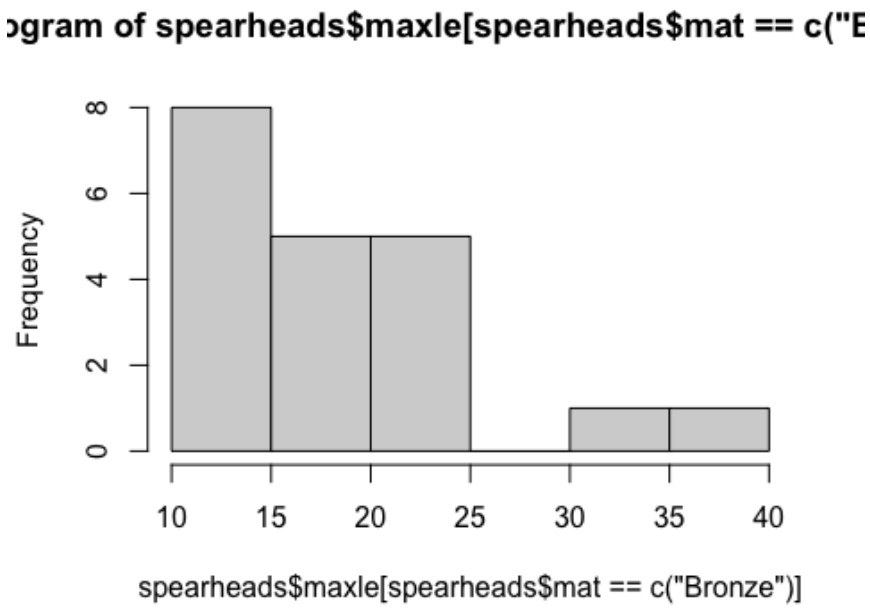


Given that our two setof data have been converted into cumulative distributions, now we can perform a non-parametric test called *Kolgomorov-Smirnov (K-S) test*. This kind of test allows ud to compare two independent samples (two-sample test). It evaluates whether two distributions differ significantly.

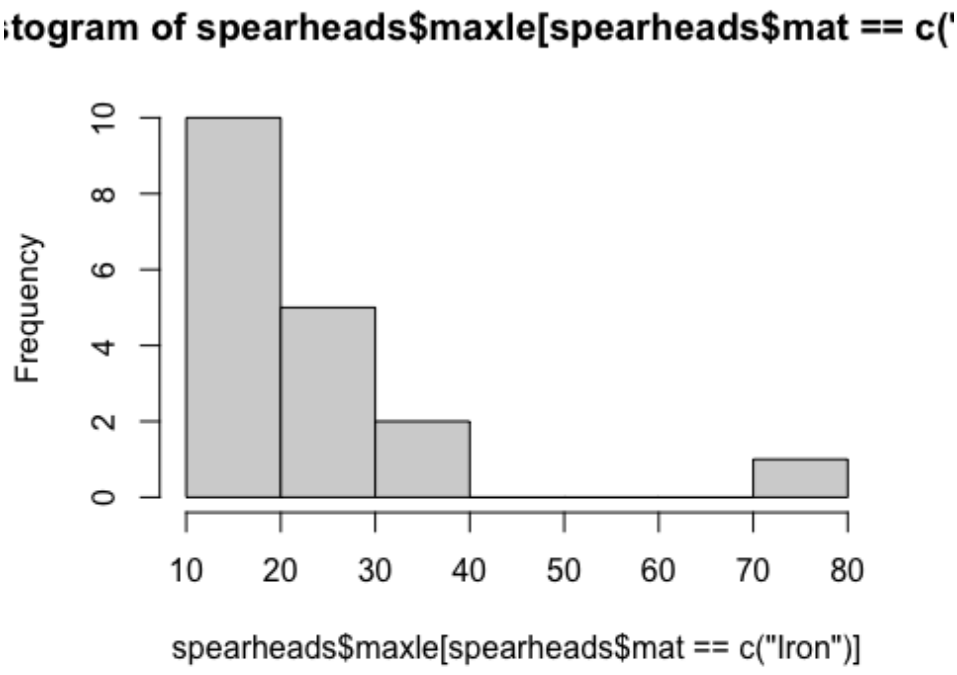
Unlike parametric methods, which rely on parameters (e.g., mean and standard deviation in a normal distribution), non-parametric methods are more flexible and can be used for data that do not follow common distributions.

Thus, before performing the *K-S test* we need to assess if the distribution of the length of Iron and Bronze spearheads is not normal.

```
hist(spearheads$maxle[spearheads$mat==c("Bronze")],breaks = 6)
```




```r
hist(spearheads$maxle[spearheads$mat==c("Iron")],breaks = 6)
```



Both distributions show a skewed distribution of values. So, the *K-S test* is suitable for this scenario.

Basically, the K-S test measures the maximum absolute difference between the empirical cumulative distribution function (ECDF) of the sample and the ECDF of the reference distribution (or another sample).

In this case, the The null hypothesis (H_0) states that the two samples come from the same distribution. A low p-value (< 0.05) suggests rejecting H_0, indicating a significant difference.

So, let us perform the test:

```
ks.test(spearheads$maxle[spearheads$mat=="Iron"],spearheads$maxle[spearheads$
mat=="Bronze"])

##
##  Exact two-sample Kolmogorov-Smirnov test
##
## data:  spearheads$maxle[spearheads$mat == "Iron"] and
spearheads$maxle[spearheads$mat == "Bronze"]
## D = 0.17778, p-value = 0.8486
## alternative hypothesis: two-sided
```

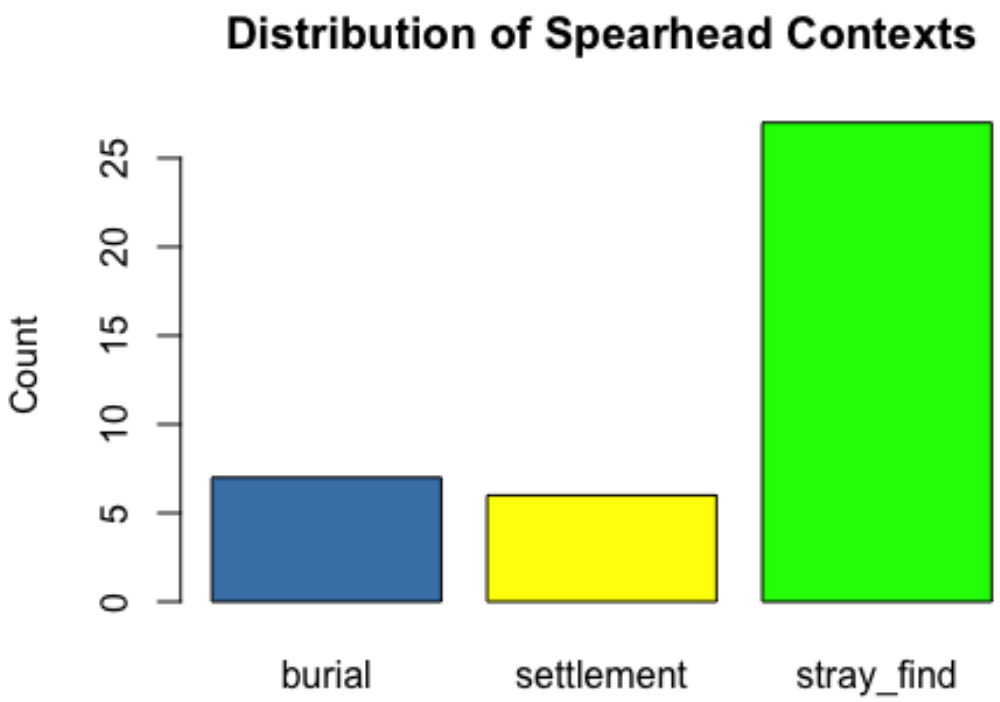
The test shows that a maximum distance in the empirical cumulative frequency distribution is 0.17778. The p-value is bigger than 0.05 and indicates that the. Bronze and Iron spearheads do not differ significantly in length.

2.4 Graphs

In this section, I will introduce you to some graphs for data visualization.

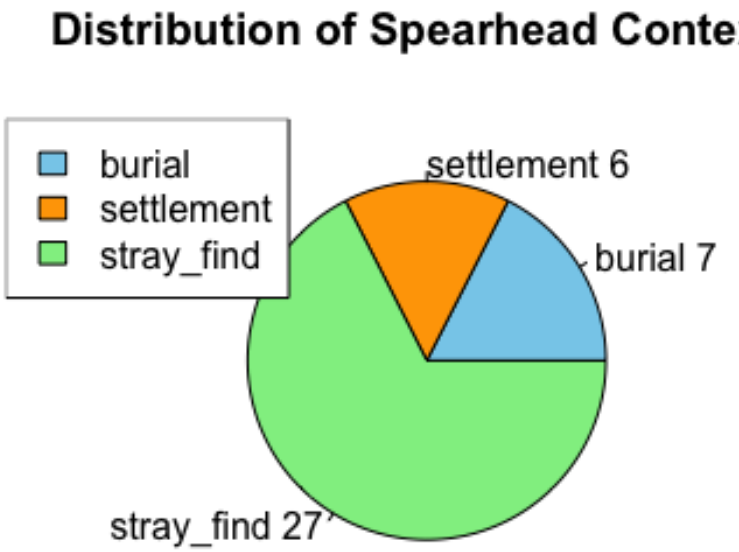
Let us create a bar chart showing the content of provenance of the spearheads:

```
# Create a frequency table for contexts
context_counts <- table(spearheads$con)

barplot(context_counts,
        main = "Distribution of Spearhead Contexts",
        col = c("steelblue", "yellow","green"),
        ylab = "Count",
        border = "black")
```



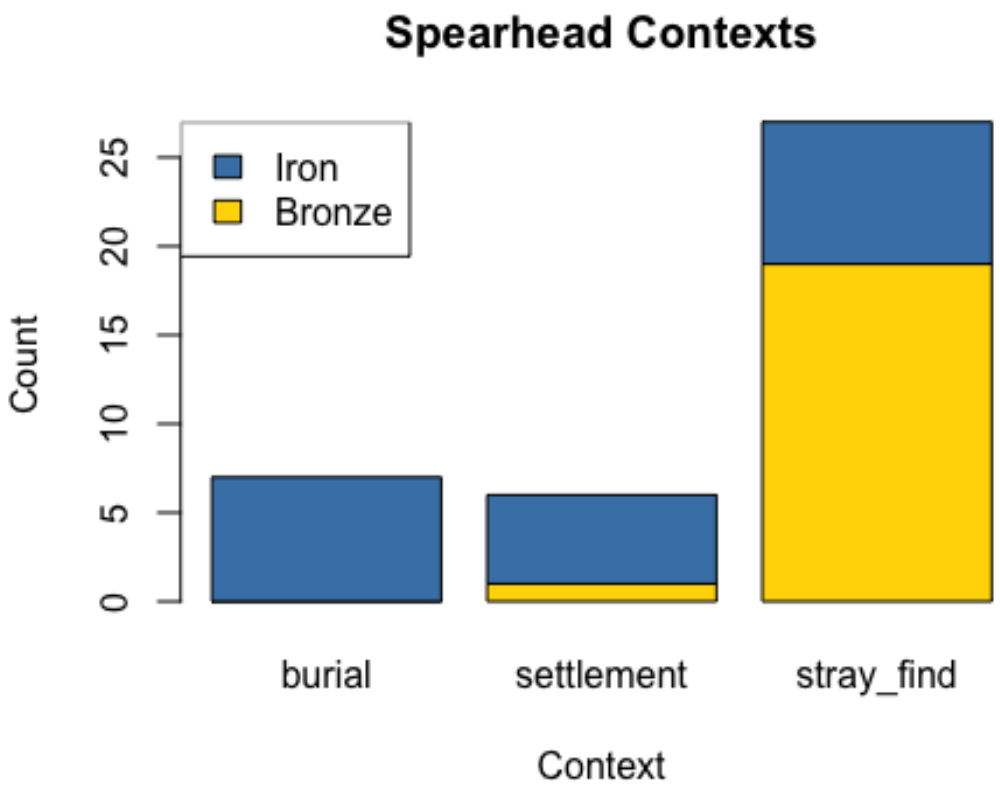
Let us try a pie chart:

```
# Create a frequency table for contexts
context_counts <- table(spearheads$con)

# Define colors for better visualization
colors <- c("skyblue", "orange", "lightgreen")

# Create the pie chart
pie(context_counts,
labels = paste(names(context_counts), context_counts),
    col = colors,
    main = "Distribution of Spearhead Contexts")

# Add a legend
legend("topleft", legend = names(context_counts), fill = colors)
```


Distribution of Spearhead Contexts
burial
settlement
stray_find
settlement 6
burial 7
stray_find 27

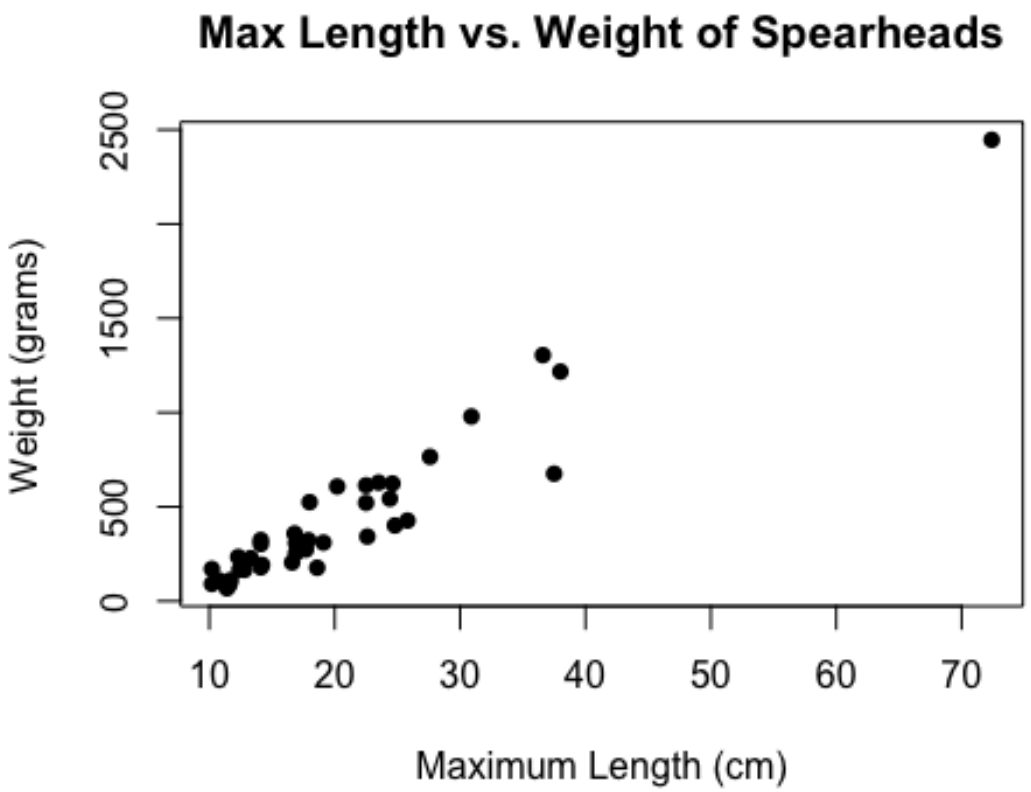
Here stacked bar chart showing where Iron and Bronze spearheads were found:

```
context_table <- table(spearheads$mat, spearheads$con)
barplot(context_table,
        main = "Spearhead Contexts",
        col = c("gold", "steelblue"),
        xlab = "Context",
        ylab = "Count")

# Add a legend
legend("topleft", legend = c("Iron","Bronze"), fill = c("steelblue", "gold"))
```



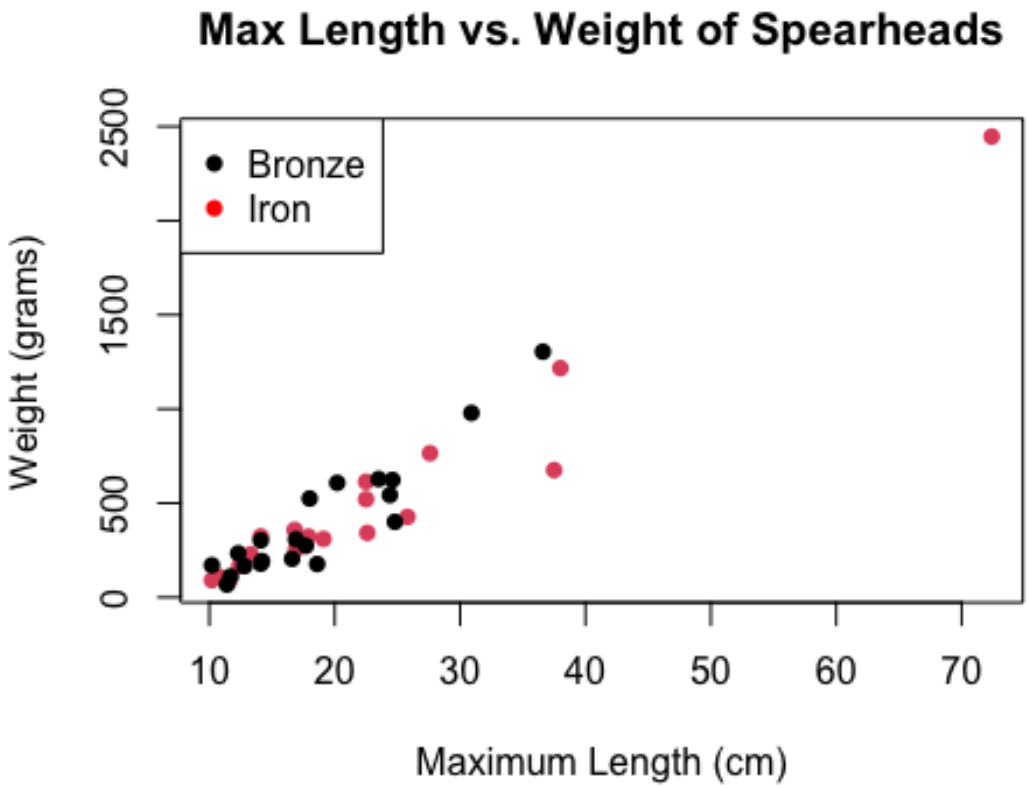
Let us plot a scatterplot showing how spearhead length affects weight:

```
plot(spearheads$maxle, spearheads$weight,
     main = "Max Length vs. Weight of Spearheads",
     xlab = "Maximum Length (cm)",
     ylab = "Weight (grams)",
     pch = 16)
```



Then we can plot the graph showing if the patterns change according to the spearheads material:

```
plot(spearheads$maxle, spearheads$weight,
     main = "Max Length vs. Weight of Spearheads",
     xlab = "Maximum Length (cm)",
     ylab = "Weight (grams)",
     col = as.factor(spearheads$mat),
     pch = 16)

#add the legend
legend("topleft", legend = levels(as.factor(spearheads$mat)),
       col = c("black","red"), pch = 16)
```



You can see that the maximum length of spearheads is positively correlated with the weight. As the length increases, the item weights more.

3. Distributions

Archaeological data can be represented using various theoretical distributions. In this section, we will focus on two widely used ones: binomial and Poisson. The binomial and Poisson distributions fall under discrete distributions, as they apply to data recorded as whole numbers, such as the count of sites, hand axes, etc.

3.1 Binomial distribution

Binomial distributions are used to model observations that belong to one of two distinct categories, such as decorated vs. undecorated, present vs. absent. One category is arbitrarily labeled as a "success," with its probability represented by p, while the probability of the other category, or "failure," is $1–p$. Based on the total number of trials or observations, it is possible to determine the likelihood of obtaining any given number of successes.

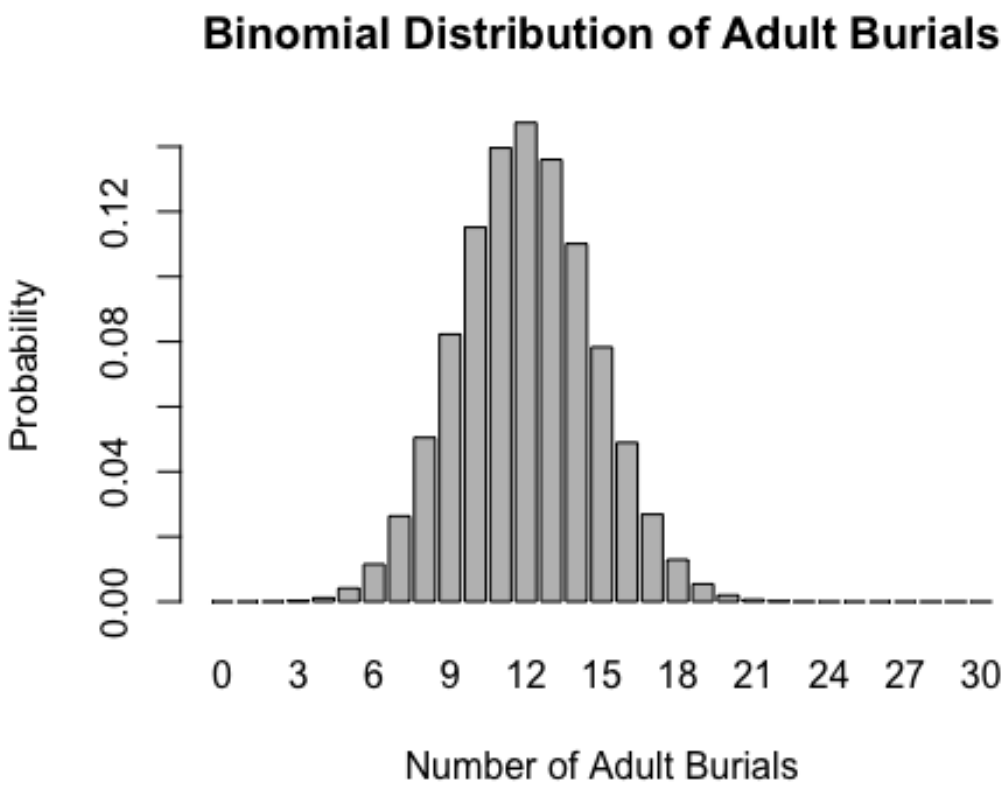
$$\Pr(X=k)=\binom{n}{k}p^k(1-p)^{n-k}$$

$$\text{where}\binom{n}{k}=\frac{n!}{k!(n-k)!};\ k=0,\ 1,\ 2,\ \ldots,\ n;$$

In a binomial distribution, *n* represents the total number of trials, while *p* denotes the probability of success. The notation **n!** (n factorial) indicates the product of all positive integers from 1 to *n* (i.e., 1×2×3×4x*n*). The right-hand side of the binomial formula calculates the probability of obtaining exactly *k* successes in *n* trials. When the probabilities for all possible values of *k* (ranging from 0 to *n*) are summed, the total equals 1. Binomial distributions occur in situations where there are only two possible outcomes. A classic example is flipping a coin, where the result can be either heads or tails, pottery can bedecorated or not, etc.

Let us imagine that we have a cemetery with 30 graves. Based on demographic data and findings from other excavations, we estimate that approximately 60% of the burials should be children, while 40% should be adults. By applying the binomial distribution, we can determine the expected number of adult graves in this cemetery. In this scenario, the probability (*p*) of success (adult burial) in each trial is 0.4.

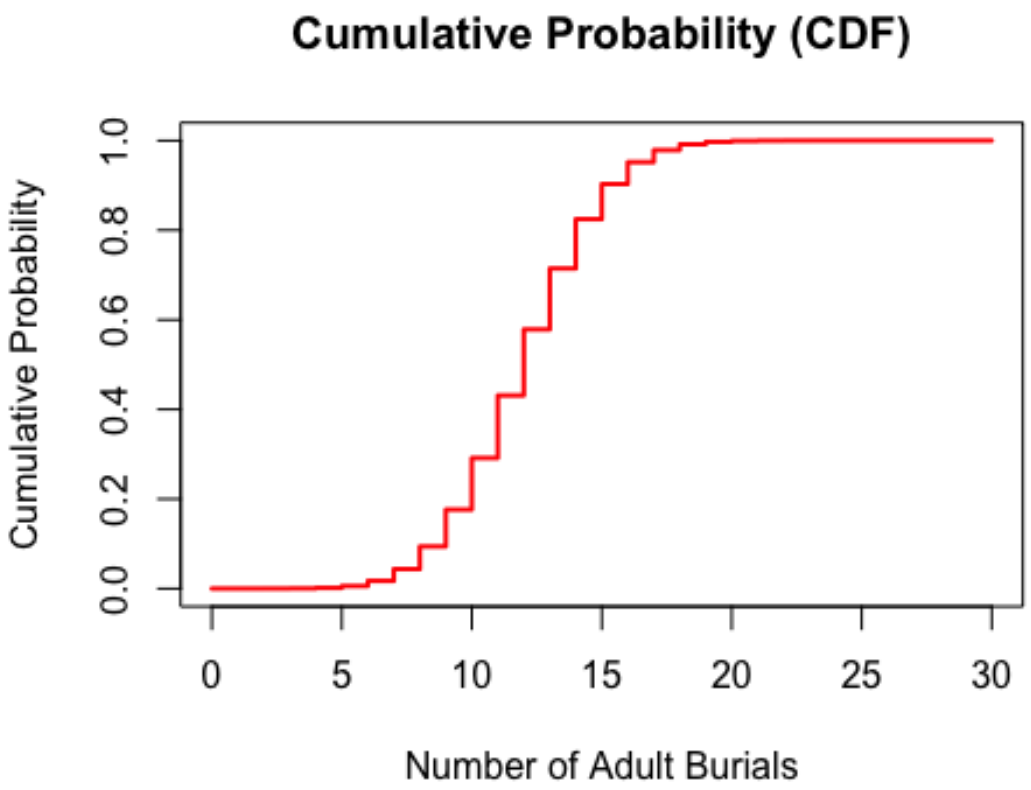
```
# Parameters
n <- 30   # Total graves
p <- 0.4  # Probability of adult burial

# Compute binomial distribution for different values of k (number of adult
burials)
k_values <- 0:30  # Possible adult counts from 0 to 30
probabilities <- dbinom(k_values, size = n, prob = p)

# Create a table of values
distribution_table <- data.frame(Adults = k_values, Probability =
probabilities)

# Plot the binomial distribution
barplot(probabilities, names.arg = k_values,
        main = "Binomial Distribution of Adult Burials",
        xlab = "Number of Adult Burials", ylab = "Probability")
```



The output gives here the probability of having exactly k adult burials for all *success* (k=0,1,2,...,30).

We can also compute the cumulative probabilities by the following command:

```
# Compute CDF (Cumulative Distribution Function)
cdf_values <- pbinom(k_values, size = n, prob = p)

# Plot CDF
plot(k_values, cdf_values, type = "s", col = "red", lwd = 2,
     main = "Cumulative Probability (CDF)",
     xlab = "Number of Adult Burials", ylab = "Cumulative Probability")
```



By looking at the cumulative probabilities above, for instance, the probability of finding from zero to 15 adult burials is then the sum of these probabilities. Hence, the probability of fnding 15 or fewer adult burials is 0.85 (or 85%). The cumulative probability of finding more than 15 adult after excavating 30 graves excavation is 1 - 0.85 = 0.15.

Continuing excavating over and over again it is possible to calculate the long-term average of the expected successes (number of adult burials) over the entire population of trials (30 graves). The expected value E(X) of a binomial distribution is given by the formula:

$E(X) = np$

So, the expected number of adult burials would be:

```
E(X) = 30*0.40. = 12
```

This means we would predict around 12 adult burials in the cemetery.

3.2 Poisson Distribution

Poisson distributions are used to model the number of successes that occur within a specific unit of time or space, where each success is independent of the others. In archaeology, the Poisson distribution is often employed as a null hypothesis in spatial

analysis. For example, the distribution of sites within a series of survey units or the number of pottery sherds in excavation squares would follow a Poisson distribution if the observations are entirely independent. However, when examining by-products of human activity, these observations are typically not independent. In such cases, it is possible to analyze whether the occurrences are evenly spread out or tend to cluster.

Te Poisson probability of any given number of the variable X can be calculated with the following formula:

$$\Pr(X = k) = \frac{\lambda^k}{k!} e^{-\lambda}$$

where k is the number of the events whose prob- ability is calculated, λ is the average number of events occurring per unit of time or space, e is a constant that is approximately equal to 2.71828, and k! is the factorial number of k.

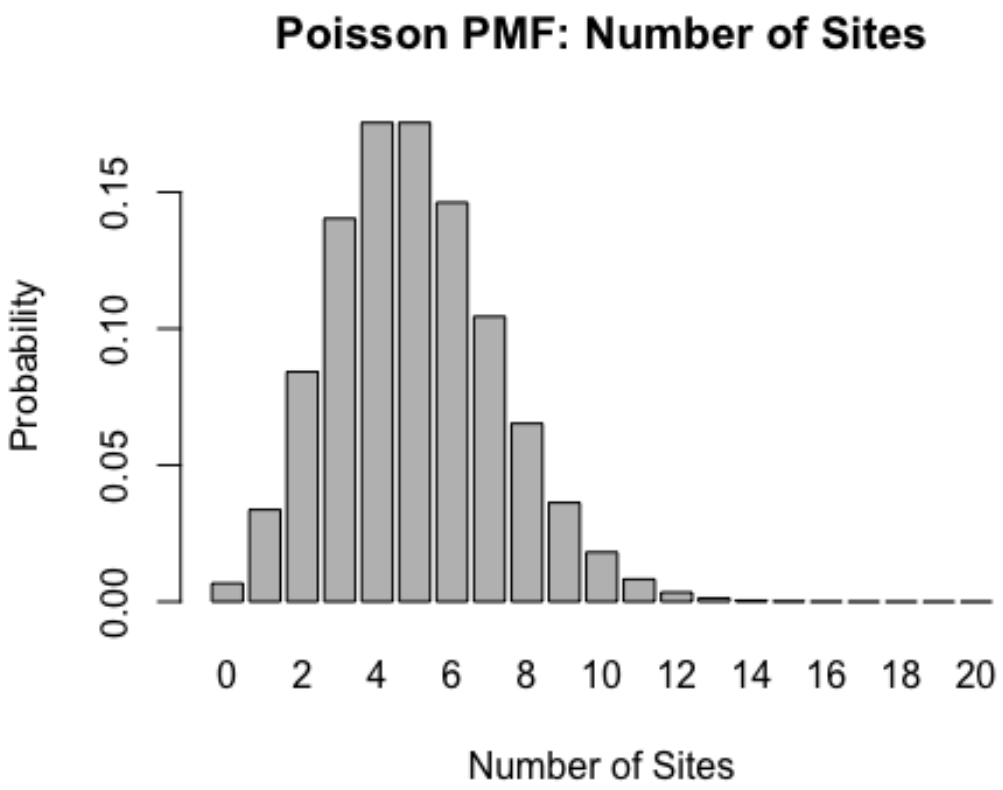
Let us imagine a scenario where during an archaeological surveys a total of 100 sites have been found on an area measuring 20 square kilometers. So, on average 5 sites were found per each one square kilometer unit.

Let us compute a Poisson distribution:

```
# Parameters
lambda <- 5  # Average number of sites per km^2
k_values <- 0:20  # Possible number of sites found within a 1 km square

# Compute PMF (Probability Mass Function) for each k
pmf_values <- dpois(k_values, lambda)

# Compute CDF (Cumulative Distribution Function) for each k
cdf_values <- ppois(k_values, lambda)
```

Plot the Poisson distribution:

```
# Plot PMF (Bar chart)
barplot(pmf_values, names.arg = k_values,
        main = "Poisson PMF: Number of Sites",
        xlab = "Number of Sites", ylab = "Probability")
```

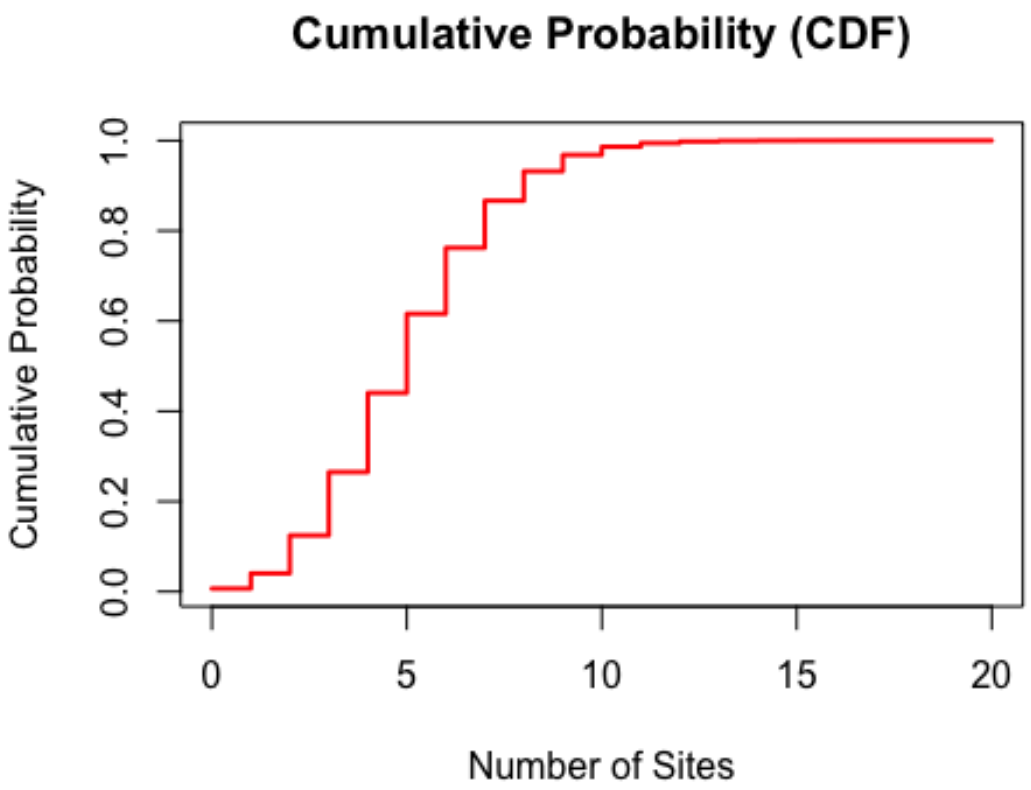


The graph above shows that is most likely finding between 3 and 6 sites within each 1 km square unit. If sites were placed independently, their distribution would follow a Poisson distribution, with most survey units expected to contain around five sites. However, if people tended to build sites near existing ones, or alternatively, away from them, the distribution would deviate from a Poisson distribution. In the first scenario, more survey units would have zero sites and a smaller number of units would have more than five sites. In the second scenario, a larger proportion of the survey units would have five or six sites.

Although the graph does not show the proabilities for values higher than twenty, this is just to save space. In fact, there is a low probability of having a square containing 20 or more sites. In the Poisson distribution it is theoretically possible to have hundreds of events within one square, even though that probability would be extremely low.

The figure below shows the cumulative probability from zero to 20 for any specifc value of *k*. This is useful to obtain the probability of finding more or fewer *k* events within a square.

```
# Plot CDF (Step line chart)
plot(k_values, cdf_values, type = "s", col = "red", lwd = 2,
     main = "Cumulative Probability (CDF)",
     xlab = "Number of Sites", ylab = "Cumulative Probability")
```



In the present example the probability of finding between one and 7 sites within a 1km square unit is 0.80.

Once all probabilities are found, then it is possible to calculate the expected number of squares that can contain *k* events by multiplying the relevant probability by the total number of squares.

Let us have a look at the poisson probabilities values:

```
pmf_values
```

```
##  [1] 6.737947e-03 3.368973e-02 8.422434e-02 1.403739e-01 1.754674e-01
##  [6] 1.754674e-01 1.462228e-01 1.044449e-01 6.527804e-02 3.626558e-02
## [11] 1.813279e-02 8.242177e-03 3.434240e-03 1.320862e-03 4.717363e-04
## [16] 1.572454e-04 4.913920e-05 1.445271e-05 4.014640e-06 1.056484e-06
## [21] 2.641211e-07
```

In this case, the expected number of squares containing five sites is 0.175 x 20 = 3.5. Comparing the expected frequencies with the observed data allows one to assess if the events are distributed randomly in a given study area.

4. Sampling

This exercise guides you through various procedures and considerations related to archaeological samples and sampling designs. It starts by exploring conventional sampling methods and non-spatial finite populations.

We can now get some summary values for the weight variable stored in this dataframe by typing:

```
summary(spearheads$weight)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    67.7   178.2   309.0   442.4   559.4  2446.5
```

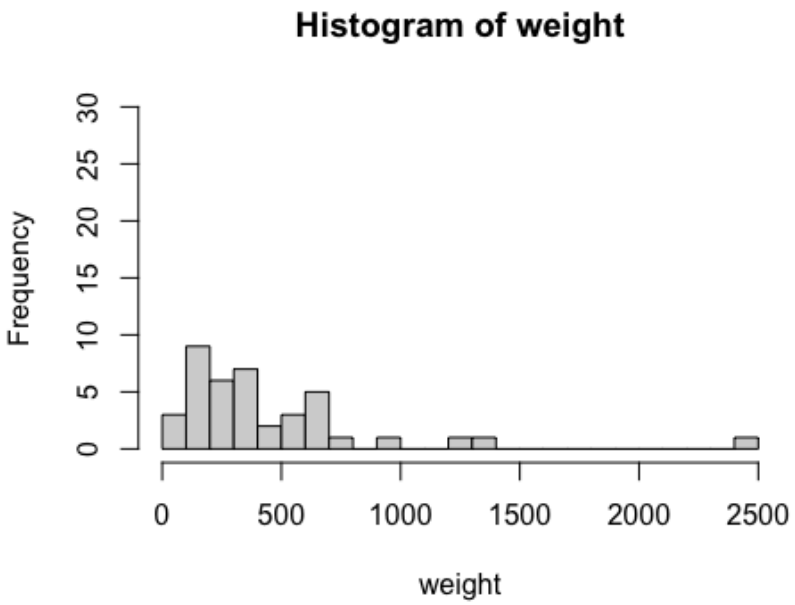
First attach the spearheads data.frame class object so that it can be more simply referred to in the commands that follow:

```
attach(spearheads)
```

n R, the `attach()` function is used to make the columns of a data frame or list accessible by their names without needing to refer to the data frame or list explicitly each time. It modifies the search path, allowing you to directly use the names of the columns in expressions or functions without needing to use the *$* operator.

Let's generate a histogram of the weight variable in this dataset, using 25 bins of 100 grams each to display the frequency distribution.

```
hist(weight, breaks = c(100*0:25), xlim=c(0,2500), ylim= c(0,30))
```



Keep in mind that you can save this plot permanently by using the `dev.print()` function, as shown below:

```
dev.print(device=pdf, file="hist_weight_all.pdf")
```

For demonstration, let's assume that these forty spearhead weights represent the entire population. We can easily compute the average (mean) weight of this population by typing:

```
mean(weight)

## [1] 442.395
```

And we can summarise the spread of the distribution of weights as being a standard by typing:

```
sd(weight)

## [1] 436.0024
```

Now, let's imagine that we don't have access to all 40 spearheads but need to estimate the mean weight based on a smaller sample. To start, we'll randomly select nine spearheads and calculate the mean weight for this sample:

```
sample.n9<-sample(weight,size=9, replace = T)
```

Notice that the sampling function is instructed to select a sample of nine values from the weight column in the spearhead dataset, with replacement. This means that if you were to take multiple samples from the dataset (as we will do below), any values selected in previous sample runs are returned to the pool and can be chosen again in subsequent samples. Alternatively, sampling without replacement is also frequently used for specific purposes.

When to Use:

1. replace = FALSE: Use this when you want to sample from a set without repeating any elements. For example, drawing lottery numbers or selecting a subset of items without repetition.
2. replace = TRUE: Use this when you want to allow repetition. For example, when drawing from a set of items with replacement (such as rolling a die or choosing items from a bag where you can pick the same item more than once).

In any case, you can now calculate the mean of the resulting sample:

```
mean(sample.n9)

## [1] 476.7889
```

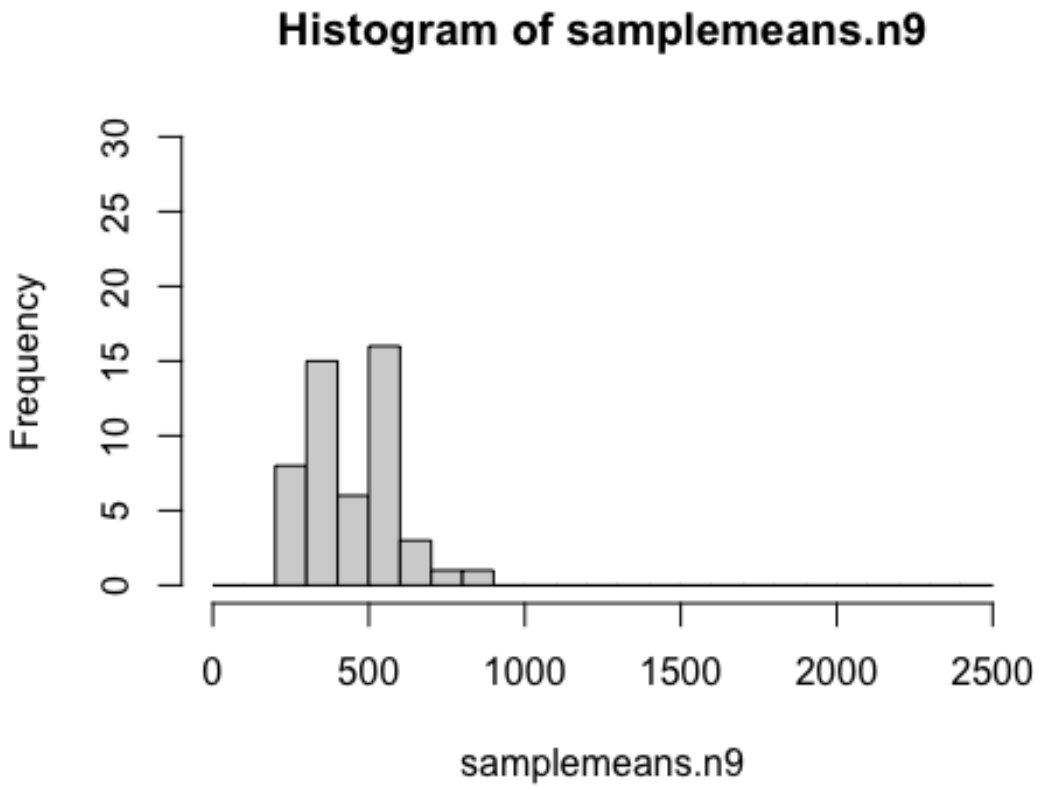
The mean value you obtain will vary depending on the specific random sample selected by the software using its random number generator. While it is likely to be a close approximation of the population mean, it is unlikely to match it exactly. To compare the

mean values obtained from multiple samples, we can repeat this process, for example, 50 times, by typing the following:

```
samplemeans.n9 <- sapply(1:50, function(i) mean(sample(weight,size=9, replace
= T)))
```

The function above selects a sample of nine spearheads, calculates the sample mean, and repeats this process for 50 separate samples. The outcome is a dataframe containing 50 mean values, which can be visualized by plotting a histogram as follows:

```
hist(samplemeans.n9, breaks = c(100*0:25), xlim=c(0,2500), ylim= c(0,30))
```

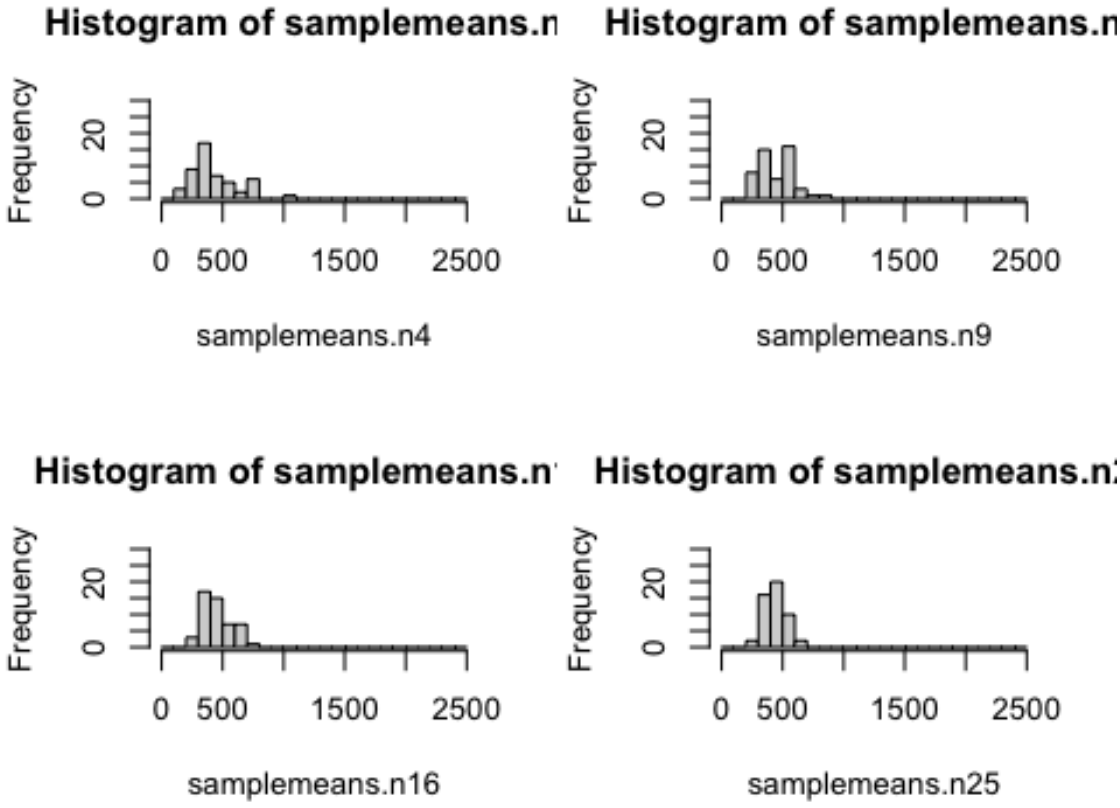


The histogram reveals that the sample means are quite consistent and cluster around the true population mean of 442.4, although there is still some variation. We can perform the same procedure for three additional sample sizes (n = 4, n = 16, and n = 25):

```
samplemeans.n4 <- sapply(1:50, function(i) mean(sample(weight,size=4, replace
= T)))
samplemeans.n16 <- sapply(1:50, function(i) mean(sample(weight,size=16,
replace = T)))
samplemeans.n25 <- sapply(1:50, function(i) mean(sample(weight,size=25,
replace = T)))
```

To conveniently visualize these three results alongside our previous one, we can create histograms for each sample size in a 2x2 layout. This can be done by first using the par(mfrow()) function and then generating the individual plots.

```
par(mfrow=c(2,2))
hist(samplemeans.n4, breaks = c(100*0:25), xlim=c(0,2500), ylim= c(0,30))
hist(samplemeans.n9, breaks = c(100*0:25), xlim=c(0,2500), ylim= c(0,30))
hist(samplemeans.n16, breaks = c(100*0:25), xlim=c(0,2500), ylim= c(0,30))
hist(samplemeans.n25, breaks = c(100*0:25), xlim=c(0,2500), ylim= c(0,30))
```



The key takeaway is that as the sample size increases, the variation around the estimate of the true mean for spearhead weight decreases (the distributions become narrower and more peaked). This observation supports the *Central Limit Theorem*, which is a fundamental concept behind most sampling methods. It describes how the distribution of sample means (or sums) tends to approach a normal distribution as the sample size increases,

Now save your plot:

```
dev.print(device=pdf,file="hist_weight_sample.pdf")
```

Due to the relationship between sample size and the variability around the sample means, we can calculate a useful statistic called the standard error. It is typically computed using the following formula:

$$s.e. = \frac{\sigma}{\sqrt{n}}$$

where σ is the sample standard deviation and n is the sample size.

Before we compute the standard error in R, let's first derive another sample of nine spearheads, but this time by setting a random seed. This seed acts as a starting point for the software's random number generator when selecting spearheads from the overall list of forty. By setting the seed, we ensure that the random numbers you generate will be the same every time you run the script, for consistency and reproducibility.

Now type:

```
set.seed(1)
sample.n9s1<-sample(weight,size=9, replace = T)
n9s1mean<-mean(sample.n9s1)
```

We can now convert equation 1 for the standard error in a R script:

```
sd(sample.n9s1)/sqrt(9)

## [1] 137.4346
```

When the sample fraction is large (i.e., when the sample proportion exceeds approximately 10%), we should adjust our standard error by multiplying it with the so-called finite population correction. This correction is formally expressed as:

$$fpc = \sqrt{\frac{N-n}{N-1}}$$

Where *N* is the population size and *n* is the sample size. We can implement all of this in R as follows:

```
se.n9s1<-sd(sample.n9s1)/sqrt(9) * sqrt((40-9)/(40-1))
```

The standard error is valuable because it enables us to compute a confidence interval for our estimated mean weight. In other words, it provides a way to quantify the uncertainty surrounding our estimate of the population mean, based on the sample size we've selected. For sample sizes smaller than approximately 40 (as described in Shennan 1997: 82-3), we use an appropriate formula to calculate the confidence interval as follows:

```
# First calculate the relevant t-score. For those who want to follow
# the detail of this calculation, the t-value is for both tails of the
# distribution hence 0.975 rather than 0.95 (see Shennan 1997: 83).
t.n9<-qt(0.975, df=9-1)
# Then calculate the confidence interval
n9conf.int<-t.n9 * se.n9s1
n9conf.int
```

```
## [1] 282.5561
```

We can then combine this with our sample mean to estimate that the population mean of all forty spearhead weights is likely to be 433.76g ± 282.55g, with 95% confidence. This means there is a 95% probability that the true population mean lies within the range of 151.21g to 716.31g.

Of course, this is a very broad range and might encourage us to try a larger sample, such as 25:

```
set.seed(1)
sample.n25s2<-sample(weight,size=25, replace = T)
mean(sample.n25s2)

## [1] 389.552

se.n25s2<-sd(sample.n25s2)/sqrt(25) * sqrt((40-25)/(40-1))
t.n25<-qt(0.975, df=25-1)
n25conf.int<-t.n25 * se.n25s2
n25conf.int

## [1] 95.52968
```

The result in this case is 389.552 g ± 95.52, which means a much lower level of uncertainty wit the true population mean within the range of 485.07g to 294.03g.

Confidence intervals are a common method for expressing uncertainty in measurements due to sampling. However, they can also be used in the opposite direction to determine the necessary number of samples needed to achieve a specific level of accuracy in estimating a population parameter (such as the mean weight). For instance, if we want to ensure an accuracy of approximately ±150 (at 95% confidence), we can calculate the required sample size using the following formula in R:

```
# The z-score for 95% confidence is 1.96 and we first calculate a
# temporary estimate of necessary sample size:
tmpsize<-((1.96*sd(weight))/150)^2
# The final estimate of the necessary sample size.
ssize.150gtol<-1/((1/tmpsize) + (1/40))
ssize.150gtol

## [1] 17.91791
```

This suggests that we would need about 18 samples to be able to estimate the population mean with an error of ±150g and a confidence of 95% that it would really fall within this range.

References

- Fletcher, M., and Lock, G., 2005. *Digging Numbers: Elementary Statistics for Archaeologists* (second edition) Oxford: Oxford University School of Archaeology

- Shennan, S. 1997. *Quantifying Archaeology*. Edinburgh: Edinburgh University Press.